Proactive Defense Against Localized Universal Adversary Attacks

BMVC 2023 Submission # 130

Abstract

This paper addresses the vulnerability of deep learning models, particularly convolutional neural networks (CNNs), to adversarial attacks and presents a proactive training technique designed to counter them. We introduce a novel volumization algorithm, which transforms 2D images into 3D volumetric representations. When combined with 3D convolution and deep curriculum learning optimization (CLO), it significantly improves the immunity of models against localized universal attacks by up to 40%. We evaluate our proposed approach using contemporary CNN architectures and the modified Canadian Institute for Advanced Research (CIFAR-10 and CIFAR-100) [11] and ImageNet[6] datasets, showcasing accuracy improvements over previous techniques. The results indicate that the combination of the volumetric input and curriculum learning holds significant promise for mitigating adversarial attacks without necessitating adversary training.

1 Introduction

The security of any machine learning model is assessed in terms of the goals and capabilities associated with adversary attacks. Algorithmically crafted perturbations, even if minuscule, can be exploited as directives to manipulate classification outcomes [2, 4, 16, 19]. Attacks can be classified as black-box or white-box[3], depending on the attacker's access to and knowledge of the model's information, which includes its architecture, parameters, training data, weights, and more. In a white-box attack, the attacker has complete access to the network's information, while a black-box attack is characterized by the absence of knowledge regarding the model's internal configuration. Occasionally, a gray-box attack can be generated by employing a generative model, enabling the creation of adversarial examples without access to the victim model. Localized adversarial attacks [15] exploit spatial invariance of CNN-based image classifiers by introducing minimal perturbations to deceive the model into producing incorrect classifications. These attacks are usually constrained to a small contiguous portion of the image and are image-agnostic (or universal) attacks.

In this paper, we introduce a new training methodology (Figure 1) designed to fortify CNNs against localized attacks. Our primary approach incorporates deep curriculum optimization [7] and a volumization algorithm. We employ an information-theoretic representation of an image along with optimization procedure that merges batch-based curriculum learning (CL), patch aggregate loss (PAL) function, and 3D convolution to train and proactively defend against effective localized attacks such as one-pixel and adversary patch attacks.

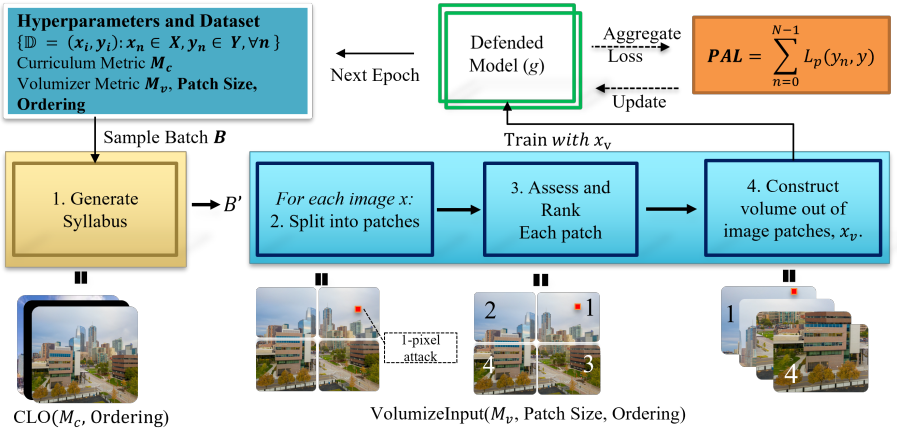


Figure 1: This figure illustrates the proposed training process. The process starts with generating a syllabus (input path) for the batch. It then volumizes each image, followed by feature extraction using 3D CNNs. The patch aggregate loss function is applied to optimize parameters.

Spatial invariance property of CNNs enables learning of features in an image regardless of their position within the image. This is achieved using convolutions on small, non-overlapping regions of the image, essentially learning local features. This property can also be exploited by localized attacks, as even tiny perturbations in specific regions can lead to incorrect classifications. To counteract this vulnerability, we designed a preprocessing algorithm that divides the input image into non-overlapping patches and generates a 3D volume of these patches for training. By focusing on smaller regions, the algorithm enables the model to learn localized features more effectively. This approach helps the models to proactively counteract localized attacks while maintaining accurate classification performance. We present results that show our method better preserves classification accuracy and has a higher defense success rate than adversary training-based defense approaches. In summary, this paper's contributions are: **(a)** A novel training methodology that employs volumization by dividing an image into patches and creating a 3D volume, combined with a curriculum learning approach for adaptive training. **(b)** Demonstrate the effectiveness of our proposed approach in defending against localized attacks through extensive experimental validation. Our results indicate that models trained using this technique exhibit improved robustness (compared to existing approaches) against N-pixel and patch attacks while maintaining high generalization performance.

In section 2, we present related work. Section 3 details problem formalism - highlighting the attack and defense objectives. In section 4 we present the proposed methodology, while the following section 5 presents experimental results and discussion. Finally we give concluding remarks in section 6.

2 Related Work

N-Pixel Attack Defenses. Su et al. uses a technique which generates adversarial examples by perturbing only one or more pixels [19], has proven to be difficult to defend. To date,

the most successful defense against this attack is a method presented by Chen et al. [4]. The authors propose Patch Selection Denoiser (PSD) that removes few of the potentially attacked pixels in the whole image. At the cost of image degradation, the authors demonstrate successful defense against one-pixel attacks. Similarly, Liu et al. [13] proposed a three-step image reconstruction algorithm to remove attacked pixels. The authors report defense success rate for N-pixel attack for N chosen from (1, 15).

Adversary Patch Defenses. Defenses for patch attacks are typically viewed as a detection problem [14, 22, 23]. Once the patch's location is detected, the suspected region would be either masked or in-painted to mitigate the adversarial influence on the image. Hayes et al. [9] first proposed DW (Digital Watermarking), a defense against adversarial patches, inspired by the procedure of digital watermarking removal [12]. A saliency map of the image was constructed to help remove small holes and mask the adversarial image, blocking adversarial perturbations. This was an empirical defense with no guarantee against adaptive adversaries. Naseer et al. [15] proposed LGS (Local Gradient Smoothing) to suppress highly activated and perturbed regions in the image without affecting salient objects. Specifically, the irregular gradients were regularized in the image before being passed to a deep neural network (DNN) model for inference. LGS could achieve robustness with a minimal drop in clean accuracy because it was based on local region processing in contrast to the global processing on the whole image as done by its counterparts. Chen et al. [5] proposed Jujutsu to detect and mitigate robust and universal adversarial patch attacks by leveraging the attacks' localized nature via image inpainting. A modified saliency map was used to detect the presence of highly active perturbed regions, which helped to place suspicious extracted regions in the least salient regions of the preserved image and avoid occlusion with main objects in the image. Jujutsu showed a better performance than other empirical defenses in terms of both robust accuracy and low false-positive rate (FPR), across datasets, patches of various shapes, and attacks that targeted different classes. Gittings et al. [8] proposed training time defense, which involves injecting adversarial vaccines into the input data to improve the robustness of the network against adversarial patch attacks. The approach achieved a defense success rate of 91.6% against adversarial patch attacks on the CIFAR-10 dataset, and was evaluated against several state-of-the-art adversarial patch attacks

The main tenet of our approach is to proactively mitigate the impact of adversarial attacks without prior knowledge of the attacks. Rather than relying on detection and removal techniques or adversary training, our method is designed to immunize models by enhancing their inherent robustness.

3 Problem Formulation

Let f be a CNN classifier, represented as $f\colon \mathbb{R}^M \rightarrow \mathbb{R}^c$, trained on a dataset $\{\mathbb{D} = (x_i, y_i) : x_n \in X, y_n \in Y, \forall n \}$ with input-label pair set (X,Y), to map a source image x (having width W, height H, number of channels C), to a set of probabilities $f(x)$. Each probability in $f(x)$ encodes the likelihood that x belongs to one of the k classes $c_i \in Y$. Here $|X| = M$ is the total number of samples in the dataset and $|Y| = c$ is the total number of classes.

Attack Objective: An attack on an image x involves adding a perturbation $r \in \mathbb{R}^m$ to x, causing the maximized class probabilities to differ between the original and perturbed images. i.e.,

$$argmax_i(f_i(x+r) \neq argmax_i(f_i(x)). \tag{1}$$

These types of attacks can be categorized as targeted or untargeted. In a targeted attack, the adversarial image $x' = x + r$ is generated to induce the classifier to assign x' to a specific target class $c_t \in Y$, where $c_t \neq argmax_i(f_i(x))$. The perturbation r is selected such that $argmax_i(f_i(x')) = c_t$. Conversely, in an untargeted attack, the perturbation is crafted to cause the classifier to assign x' to any incorrect class without a particular target. In this case, the perturbation r is chosen to satisfy $argmax_i(f_i(x')) \neq argmax_i(f_i(x))$ without imposing additional constraints on the target class. In this study, we only focus on untargeted attacks.

Defence Objective: The defense objective is to train a model g that is robust to untargeted universal adversarial image attacks without sacrificing the accuracy of the classifier on the original dataset. Formally, the objective is to find g that minimizes the following loss:

$$min_g \frac{1}{|D|} \sum_{(x,y)\in D} \max_{r\in R} L(g(x+r), y) \tag{2}$$

where R is the set of possible adversary perturbations, and L is a loss function used to train the model g. The objective is to minimize the maximum loss over all possible adversarial examples x' generated by any allowable perturbation in $r \in R$. In this study, R is constrained to be a set of localized attacks. Localized attacks are characterized by the property that the L^2 norm of the perturbation vector r, denoted by $||r||$, is much smaller than that of the original input image x, denoted by $||x||$ (i.e. $||r|| \ll ||x||$). These attacks modify only a small subset of pixels that are confined to a localized region of the image.

4 Method

Our aim is to develop a defended classifier, g, that inherently defends against localized attacks. To this end we propose a proactive defense approach that incorporates the following key steps: **(1)** Volumization - for each image in the batch, we convert it to a 3D volume (section 4.1); **(2)** 3D Convolution - we modify contemporary CNN architectures for 3D convolution, which enables the model to extract features from the 3D volumes. Details of the modifications are discussed in section 4.2; **(3)** Deep curriculum learning optimization - for each batch taken from the dataset D, we generate a syllabus that determines the input order of the samples in the batch. This curriculum learning approach helps the model learn progressively from simpler to more complex samples (section 4.3).

This approach ensure that the model remains resilient to perturbations and maintains accurate classification and verification of both the original images x and adversarial images x' . By converting images to 3D volumes, our method can capture and preserve spatial information that can help in defending against localized attacks. The deep curriculum learning optimization further improves the model's ability to recognize and learn from complex patterns, which aids in the proactive defense. Figure 1 depicts the end-to-end training process.

4.1 Volumization Algorithm

The volumization algorithm, is a pixel-preserving transformation operator denoted as a function:

$$V : (x,\ H,\ W,\ C) \rightarrow \left(x'_i,\ p_h,\ p_w,\ C,\ N\right). \tag{3}$$

Given an input image $x \in D$ of shape *(H, W, C)*, the algorithm uses a configurable hyperparameter, patch size $P(p_h,\ p_w)$ - where p_h and p_w are the height and width of each

patch - to split x into N non-overlapping patches x'_i of size P satisfying the pixel conservation condition:

$$x = \bigcup_{i=0}^{N-1} x'_i \tag{4}$$

This states that the original image x is equivalent to the union of all its extracted patches. Here, $\cup$ denotes the union operation. The algorithm extracts a list of patches and proceeds to rank each patches according to a prespecified metric m – a configurable hyperparameter. The chosen metric can be of type distance or standalone (Table 1). If m is a distance-based metric, a reference patch P_{ref} is selected, which can be user-defined or automatically determined by choosing the patch with the lowest entropy. On the other hand, m is considered standalone if it measures some characteristics of a given patch. All patches are then ordered based on their individual metric scores or their distances from the reference patch. The ordering *ord* is user-define configurable hyperparameter that can be either descending or ascending. Finally, the ranked patches are stacked along the depth axis to create a 3D volume $x_v = V(x)$ of shape $(p_h,\ p_w,\ C,\ N)$, where N (depth of the volume) is the total number of patches. Refer to section 4.3 for detail on the ranking and ordering process, which is identical for both the volumizer and CL when generating a syllabus for a batch.

4.2 Architecture Modifications

Given a conventional CNN architecture f, designed to learn from 2D images, we perform the following modifications to construct g - a 3D counterpart of f. The modifications result in a significant but tolerable increase in the number of parameters for each model, with VGG16 increasing by 20,663,968 parameters, ResNet50 by 48,566,533 parameters, InceptionV3 by 44,252,266 parameters, and EfficientNetB0 by 10,569,436 parameters.

Input Layer: The input layer of f, denoted as I_{2D}, is designed to accept a 2D input data x with dimensions $H \times W \times C$ such that $I_{2D} : x \in R^{H\times W\times C}$. In order to extract features from the volumized images, the input layer of model g is adjusted to be 3-dimensional,I_{3D}, where the input to this layer x_v is a 4D tensor with dimensions $p_h \times p_w \times C \times N$, such that $I_{3D} : x_v \in R^{p_h\times p_w\times C\times N}$. In short hand notation, this modification can be represented as,

$$f(I_{2D} : x \in \mathbb{R}^{H\times W\times C}) \rightarrow g(I_{3D} : x_v \in \mathbb{R}^{p_h\times p_w\times C\times N}) \tag{5}$$

Convolution Layers: In a CNN, convolution represents the interaction between an input (image or feature map) and a kernel K. K is a small matrix that slides over the input data, performing an element-wise multiplication and summation of the results to generate a feature map. In a 2D convolution, both the input data and the kernel are two-dimensional.

$$(K * x)(i, j) = \sum_m \sum_n K(m,\ n)\ x(i-m,\ j-n) \tag{6}$$

Here, K represents the 2D kernel, x represents the 2D input, and $(i,\ j)$ are the coordinates in the output feature map. The summation is performed over all spatial dimensions $(m,\ n)$ of the 2D kernel.

A 3D counterpart of the above operation is:

$$(K_{3D} * x_v)(i, j, k) = \sum_{m'} \sum_{n'} \sum_{p'} K_{3D}(m', n', p') x_v(i-m', j-n', k-p') \tag{7}$$

where K_{3D} represents the 3D kernel, x_v is the 3D input data, and (i, j, k) are the coordinates in the output feature map. The summation is performed over all spatial dimensions (m', n', p') of the 3D kernel. This modification enables the classifier to learn features from the volumized data by processing spatial information across height, width, and depth dimensions simultaneously.

Note that the optimal kernel size depends on the size of the individual patches within the volume and the desired level of spatial information capture. For example, if f consists of 1x1, 3x3, and 5x5 2D convolution layers, we adjust these layers to be 1x1xN, 3x3xN, and 5x5xN 3D convolution layers, respectively. Here, N is the depth of the input volume x_v containing patches of size (p_w, p_h). To ensure compatibility, we enforce the constraint that the kernel size is much smaller than the size of the individual patches and that the kernel operates on each patch in the volume. That is, $m' \ll p_w$, and $n' \ll p_h$. The stride and padding values are also adjusted accordingly.

Pooling Layers: All pooling layers of f are modified to handle the 3D volume representation of the input data. In f, the 2D pooling layers are denoted as P_{2D}:

$$P_{2D} : \mathbb{R}^{(H_{in} \times W_{in} \times C_{in})} \rightarrow \mathbb{R}^{(H_{out} \times W_{out} \times C_{out})} \tag{8}$$

where H_{in}, W_{in}, and C_{in} represent the height, width, and number of channels of the input image or feature maps, while H_{out}, W_{out}, and C_{out} denote the height, width, and number of channels of the output feature maps, respectively.

To effectively process volumized inputs, we replace the 2D pooling layers with 3D counterparts:

$$P_{3D} : \mathbb{R}^{(p_{h_{in}} \times p_{w_{in}} \times C_{in} \times N)} \rightarrow \mathbb{R}^{(p_{h_{out}} \times p_{w_{out}} \times C_{out} \times N)} \tag{9}$$

where $p_{h_{in}}$, $p_{w_{in}}$, and C_{in} represent the height, width, and number of channels of the input volume, while $p_{h_{out}}$, $p_{w_{out}}$, and C_{out} denote the height, width, and number of channels of the output 3D volume, respectively. N represents the number of patches.

Pooling layers reduce the spatial dimensions of the input data while preserving essential features. For example, 3D max pooling computes the maximum value of the elements within the pooling window, effectively preserving the most salient features while reducing dimension of the input. In the case of 3D max pooling, the operation is performed not only across height and width but also across depth, ensuring that the most important features in each patch are retained.

The inclusion of the depth dimension in 3D pooling allows for better preservation of spatial relationships in the input data, thus preserving more contextual information during the down-sampling process. This has significant implications considering the model's robustness against adversarial attacks. When using 2D pooling, the model may be more susceptible to adversarial perturbations that exploit weaknesses in the spatial relationships between image regions. Conversely, 3D pooling preserves the spatial relationships between regions, allowing the model to better understand and leverage the contextual information present in the input image. This is because 3D pooling operation is applied across not only width and height, but also the depth dimension, which includes the different patches. This takes into account the spatial relationships between the patches, preserving more contextual information during the down-sampling process. For instance, 3D max-pooling would select the maximum value within its 3D pooling window, which spans across multiple patches, and retain the spatial relationships between them.

Fully Connected and Output Layers: We modify f's fully connected layer to be $FC : \mathbb{R}^{(M,\ N)} \rightarrow \mathbb{R}^{(L,\ N)}$, where M represents the number of input features, L denotes the number of output features, and N is the total number of patches. This maps each patch's input features to its respective output features. Note that, to ensure compatibility with the FC layers, the outputs features M of the preceding layers must be reshaped or flattened. The output layer function, O, maps output features L of FC to k number of classes: $O : \mathbb{R}^{(L,N)} \rightarrow \mathbb{R}^{(k,N)}$. This modification allows g to map each patch's feature to its respective class probabilities, thereby enabling patch-wise error L_p calculations during training.

Patch Aggregate Loss (PAL) Function is designed to enable backpropagation on individual patches. During training, the loss for each patch is calculated separately. The patch-wise losses are then aggregated to obtain the overall loss for the image. Given the modified output O, we compute PAL in two steps as:

1. **Patch-wise Error Calculation**. We first compute the loss for each patch separately using a suitable loss function $L_p : \mathbb{R}^{(\ N',\ k)} \rightarrow \mathbb{R}^{(N')}$. Given a patch $n \in \{0,\ 1,\ ...,\ N-1\}$, the patch-wise loss is calculated as $L_p(y_n, y)$, where y_n represents the predicted class probabilities of path n, and y denotes the true class labels of the originals input.

2. **Loss Aggregation**. At this step we combine the patch-wise losses to obtain the overall loss for the image. PAL is determined by summing up the individual patch-wise losses:

$$PAL\ =\ \sum_{n=0}^{N-1} L_p(y_n,\ y) \tag{10}$$

Using the sum of patch-wise losses directly emphasizes the importance of minimizing the error for each individual patch, driving the model to learn more robust features from each patch. This increased emphasis on localized features results in a more robust model that is better equipped to counteract attacks.

4.3 Training Methodology

We incorporate curriculum learning optimization (CLO) at a batch level to enhance the training process. Given a batch $B \subseteq \mathbb{D}$, we define a syllabus $\mathbb{S}$ as a function $\mathbb{S} : B \rightarrow B'$, where B' is a reordered version of the original batch B. $\mathbb{S}$ describes an input order of the samples in B' such that the learning process progresses from simpler to more complex samples as quantified by a concrete metric m taken from Table 1. Ordering of samples for the batch

Metric	Category	Implementation
Entropy (H)	standalone	$\sum_{i \in \chi, x \in T} b_x(i) \log \frac{N}{b_x(i)}$
Mutual Information (MI)	distance	$H(x_1) + H(x_2) - JE(x_1, x_2)$
KL-Divergence (KL)	distance	$D_{k\|\|L}(x_1, x_2) = \sum_i x_{1_i} \log \frac{b_{x_1}(i)}{b_{x_2}(i)}$
Peak Signal to Noise Ratio (PSNR)	distance	$PSNR = 20 \log_{10}(\frac{MAX}{\sqrt{MSE}})$
Max Norm (MN)	distance	$x_\infty = \max(x_1, \ldots, x_n)$
Joint Entropy (JE)	distance	$JE(x_1, x_2) = \sum_i b_x(i) \log b_x(i)$
Mean Squared Error (MSE)	distance	$\frac{1}{N^2} \sum_i^N \sum_j^N (x_{1_{ij}} - x_{2_{ij}})^2$

Table 1: List of measures used in this study. Given samples x, $x_1, x_2 \in B$ where b_x is normalized histogram of pixel intensities and i is an index of a pixel value in the image's vector.

is done in the same way the volumizer algorithm orders patches to create a volume. Given $B = \{x_1, x_2, \ldots, x_n\}$, a batch of n samples (or a set of patches belonging to x), let $\mathrm{SM}(x_i)$ be the standalone metric value of x_i and $DM(x_i, r)$ be the distance metric value of the sample or patch x_i with respect to a reference image (or patch) r, respectively. We define order relations $R_{\mathrm{SM}} \subseteq B$ and $R_{\mathrm{DM}} \subseteq B$, such that:

$$(x_i, x_j) \in \begin{cases} R_{SM} \; if \;\; SM(x_i) \leq SM(x_j) \\ R_{DM} \; if \;\; DM(x_i, r) \leq DM(x_j, r) \end{cases} \tag{11}$$

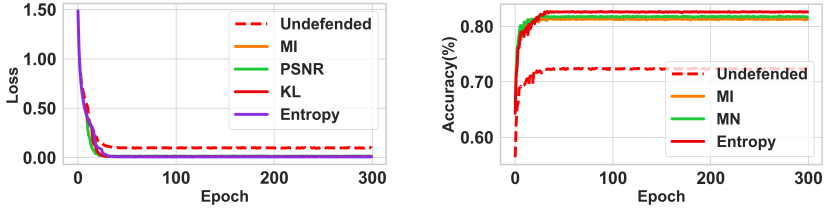
Thus, the syllabus (or volumizer) algorithm transforms B (set of patches) into an ordered one B':

$$S_{SM}(B) = \{x'_1, \ldots, x'_n\}, \; where \; (x'_i, x'_j) \in R_{SM}\} \tag{12}$$

$$S_{DM}(B) = \{x'_1, \ldots, x'_n\}, \; where \; (x'_i, x'_j) \in R_{DM}\} \tag{13}$$

This ordering ensures a progression from simpler to more complex samples based on the chosen metric during the learning process. By ordering the batch based on the complexity of images as measured by a concrete metric, the training process adapts to the dataset's inherent structure, allowing the model to adjust its learning strategy as it encounters increasingly complex samples. This is proven to result in better model performance and faster convergence [7]. In addition, this approach enables the models to becomes more robust against low-level, localized attacks. Since the patches are ordered and positioned within the volume based on their characteristics, adversarial perturbations in a specific patch are less likely to affect the overall understanding of the image by the model.

Figure 2 presents the losses and success rates on CIFAR10 for both g and f. We observe that during the 300-epoch training phase of g under varying syllabi, its loss approaches zero while its defense success rate rises to above 80%. This indicates that the proposed approach effectively defends against 1-pixel attacks introduced at each validation run. Once the classifier is trained for at least 50 epochs, it rapidly learns not to be deceived by the attacked pixel, resulting in increased success rates for g, while those for f stagnate below 72%. This finding confirms that the approach generates models that perform on par with the undefended model on clean datasets while defending against localized attacks.



(a) Training Losses (b) Training Success Rates

Figure 2: CIFAR10 training losses and success rates of defence on EfficientNet. (a) shows the losses of defended model g using different Syllabus configurations and undefended model f. (b) shows the training success rates of both models under 1-pixel attack.

5 Experiment and Discussion

We evaluate the approach on CNN models trained using popular and contemporary CNN architectures; EfficientNet-B0 [21], InceptionV3 [20], ResNet50 [10] and VGG16 [18]. We convert the open-source implementation of each model to a 3D-input-compatible architecture using the modifications highlighted above. All models are implemented using the opens-source TensorFlow [1] library. We used FoolBox[17] implementations of N-pixel attacks. **Attack Strategy:** We consider N-pixel attacks, where N is taken from the interval $[1,\ 2 \times p_w \times p_h]$ and patches of size up to 25% of the total image area. During validation, the attack is constrained to 4 regions (Figure **??**). If the attacked region size spans multiple quadrants, the attack is placed at the center of the image. The average of the five attacked classifications results is then considered the final classification under attack setting. The attacked pixels from each quadrant are selected at random. This strategy ensures a focused and fair evaluation of our approach and provides a comprehensive assessment of its robustness. **Datasets:** We compared the performance of the original model f and a defended model g on CIFAR10, 100 and ImageNet benchmark datasets under different attack settings. We use CIFAR10 to do comprehensive study and CIFAR100 and ImageNet to assess the generalizability of the approach. We compare our results to baseline defense approach that uses similar datasets. **Metrics:** We measure the classification accuracy of the models on both clean images (acc_{clean}) and adversarial images (acc_{attack}). We calculate the robustness score (δ) as the difference between the model's classification accuracy on adversarial images and its classification accuracy on clean images, $\delta = (acc_{\text{attack}} - acc_{\text{clean}})$. A larger δ demonstrates greater proactive robustness. We also measure the defense success rate (β) — the percentage of successfully defended attacks. To calculate the defense success rate, we attack a portion of the images in the test set, while the robustness score is calculated using all images under attack. A higher defense success rate indicates a better proactive defense both in experimental settings and real-world scenarios.

Defense effectiveness: We evaluate the performance of our defense in reducing the effectiveness of N-Pixel and APA attacks. We used 1, 2, up to 16-pixel coverage for N-pixel. We use adversarial patches – Toaster, School-Bus, Lipstick and Pineapple - synthesized by attack methods A-ADS [2], covering upto 25% of the entire image. Our approach is compared with existing defense strategies in terms of clean accuracy and attack accuracy or defense success rate. We mount such attacks against our defence (MI, KL, Entropy, and PSNR syllabi), and an undefended model as a control.The patch size $P(p_h,\ p_w)$ of the volumization algorithm for all syllabi is set to 16x16 pixels. We take reported results of all baseline defenses mentioned in section 2 for comparison.

Figure 3 illustrates the overall robustness (δ) of EfficientNet 3(a) and Inception 3(b) against N-pixel and patch attacks, respectively. The plots highlight the dependence of model robustness on attack size for both defended and undefended models, with the undefended model being 40% less accurate at worst. Our defense is effective for both architectures at all attack magnitudes. However, similar to the undefended model, the performance of our method degrades as the size of the attack increases, indicating a shared vulnerability to larger-scale attacks.

As depicted in Table 2(top), our proposed defense demonstrates a significant improvement on N-Pixel attack accuracy compared to the undefended models. The undefended models exhibit a sharp decline in Acc_{attack}. compared to Acc_{clean}. Our proposed method (MI) not only achieves high Acc_{clean} of 96.3% for EfficientNet and 99% for VGG and Inception but also significantly improves Acc_{attack} values to 91.3% for EfficientNet, 98.6% for VGG,

Method	Acc_{clean}			Acc_{attack}		
	Effn	VGG	Inception	Effn	VGG	Inception
Undefended	98	98.9	99	44.3	35.9	12.3
MI/Ours	**96.3**	**99**	**99**	**91.3**	**98.6**	**96.12**
PSD	-	99.53	-	-	97.8	-
Liu et al.	-	89.8	-	-	91	-

Method	Defense Success Rate (α)		
	CIFAR10	CIFAR100	ImageNet
H/Ours (Effn)	91.3	80.5	-
MI/Ours (ResNet)	95.6	**95.43**	**89.1**
PSNR/Ours (Inception)	**96.12**	**94.3**	83.2
Jujutsu (ResNet)	86.5	55.7	-
LGS	93.2	73.7	-
Vax-a-Net(VGG)	-	91.6	86.8
DW(VGG)	-	-	65.2
DW(Inception)	-	-	66.2

Table 2: (Top) Accuracy of models over the set of test images without attacks Acc_{clean} and with attack Acc_{attack}, reported for all CIFAR10 classes. Reported as top-1 accuracy of 1 pixel attack for the undefended model, the model defended by our method (MI) and other comparable approaches. All images in the dataset are attacked for this report. (Bottom) Defense success rate (α) of various defense methods against APA covering at least 5% of the image. Adversary patches; toaster, lipstick, pineapple, and school-bus were used.

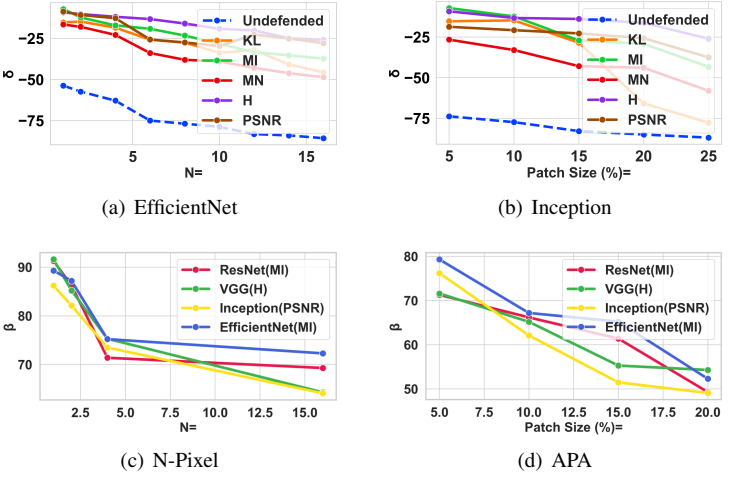
and 96.12% for Inception. Comparing MI with the PSD method, our approach has a slightly lower Acc_{clean} for VGG, with a difference of 0.53%, but delivers a better Acc_{attack} for the same model, with an improvement of 1.2%. When comparing MI to Liu et al.'s method, our method demonstrates a substantial improvement in Acc_{clean} for VGG, with a difference of 9.2%, and a higher Acc_{attack} as well, with an improvement of 4.6%. Notably, PSD and Liu et al. do not provide results for EfficientNet and Inception.

Table 2(bottom) presents defense success rates (α) for various defense methods and ours against APA on three datasets. For CIFAR10, our methods achieved 95.6% and 96.12% success rates, while Jujutsu and LGS obtained only 86.5% and 93.2%, respectively. Similarly, for CIFAR100, our methods reached success rates of 95.43% and 94.3%, outperforming Jujutsu's 55.7% and LGS's 73.7%. In the ImageNet dataset, MI/Ours achieved the highest defense success rate of 89.1%, while PSNR/Ours obtained 83.2%, both surpassing Vax-a-Net's 86.8% and DW's 65.2% and 66.2% for VGG and Inception models, respectively. Not all methods have reported results for every dataset, limiting a comprehensive comparison of their effectiveness.

Network Architecture and attack size: Figures 3(c) and 3(d) depict the impact of attack size on various CNN architectures, revealing that the defended models are highly resilient – compared to the undefended model – as the attack magnitude increases. However, once the attacked pixels surpass the individual patch size of the volume, the defense effectiveness begins to wane. This behavior can be attributed to the fact that the volumization algorithm counters small size attacks. When the perturbation size exceeds the patch size or if it spans more than one patch in the volume, the algorithm's ability to isolate and counteract the adversarial noise diminishes.

Timing information: Table 3(e) compares the time taken for inference using our method and undefended. An inference pass on the defended model takes about 6 milliseconds more time compared to the undefended. The overhead is primarily due to the volumization al-

gorithm applied at preprocessing stage. However our defence does incur significant more overhead during training. It can incur hours to days depending on the size of the dataset. This process only needs to be run once, as does training the model a priori. All inference runs used NVIDIA RTX A4000 GPU and training was done on a multi-gpu cluster.



(a) EfficientNet (b) Inception

(c) N-Pixel (d) APA

Method	VGG	ResNet	Inception
Undefended	0.12	0.14	0.09
MI/Ours	0.23	0.18	0.16

(e) Inference Time

Figure 3: (a) EfficientNet robustness as a function of N - number of pixels attacked, (b) Inception robustness against APA as a function of patch size, (c) Defense success rate (β) of various models as a function of N-Pixel attack magnitude, (d) Defense success rate (β) of various models as a function of APA attack magnitude, (e) Inference time for VGG, ResNet, and Inception, (f) Training time vs Epoch for an undefended model trained on ImageNet and the same model with our defense based on MI syllabus.

6 Conclusion

We introduced a proactive defense approach against localized adversarial attacks, which preserves model performance on clean data. Our method combines a volumization algorithm that converts 2D images into 3D volumetric representations while maintaining spatial relationships, increasing resilience to perturbations. Additionally, we employ a deep curriculum learning optimization strategy, ordering training samples by complexity, enabling progressive learning from simple to complex samples. By incorporating these techniques into popular CNN architectures, we demonstrated the effectiveness of our method against N-pixel and patch attacks. Experimental results indicated improved robustness without sacrificing performance on clean data, confirming our approach's ability to enhance image classification model resilience against localized adversarial attacks.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous systems. https://tensorflow.org, 2015.

[2] Tom B Brown, Dandelion Man'e, Aurko Roy, Mart'in Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2018. URL http://arxiv.org/abs/1712.09665.

[3] Anirban Chakraborty, M Alam, Vikram Dey, Ansuman Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018. URL http://arxiv.org/abs/1810.00069.

[4] Dongxian Chen, Ruiqi Xu, and Bo Han. Patch selection denoiser: An effective approach defending against one-pixel attacks. In *Neural Information Processing*, pages 286–296. Springer, 2019. doi: 10.1007/978-3-030-36802-9_31.

[5] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Hopskipjumpattack: A query-efficient decision-based attack. *arXiv preprint arXiv:1904.02144*, 2019.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[7] Hailay Ghebrechristos and Gita Alaghband. Deep curriculum learning optimization. *SN Computer Science*, 1(5):245, 2020. doi: 10.1007/s42979-020-00251-7.

[8] Thomas Gittings, Stephen Schneider, and John Collomosse. Vax-a-net: Training-time defence against adversarial patch attacks. In Hideo Ishikawa, Cheng-Lin Liu, Tomas Pajdla, and Jianbo Shi, editors, *Computer Vision–ACCV 2020*, volume 12625, pages 235–251, Cham, 2021. Springer International Publishing. doi: 10.1007/978-3-030-69538-5_15.

[9] Jamie Hayes. On visible adversarial perturbations digital watermarking. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1678–16787, Salt Lake City, UT, USA, 2018. doi: 10.1109/CVPRW.2018.00210.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf. p. 60.

[12] Wei Liu, Jiajia Jiang, Yajie Zhang, and Yi Yang. Deep learning for generic watermark removal. In *Proceedings of the 2018 ACM on Multimedia Conference*, pages 1336–1344. ACM, 2018.

[13] Zhi-Yi Liu, Po-Shen Wang, Shih-Chieh Hsiao, and Ricky Tso. Defense against n-pixel attacks based on image reconstruction. In *Proceedings of the 8th International Workshop on Security in Blockchain and Cloud Computing*, pages 3–7, New York, NY, USA, 2020. doi: 10.1145/3384942.3406867.

[14] Jan Hendrik Metzen, Tim Genewein, and Volker Fischer. Detecting and defending against adversarial attacks on neural networks. In *International Conference on Learning Representations*, 2017.

[15] Muzammal Naseer, Salman Khan, and Fatih Porikli. Local gradients smoothing: Defense against localized adversarial attacks. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1300–1307, 2019. doi: 10.1109/WACV.2019.00143.

[16] Srijan Rao, David Stutz, and Bernt Schiele. Adversarial training against location-optimized adversarial patches. In Hideo Ishikawa, Cheng-Lin Liu, Tomas Pajdla, and Jianbo Shi, editors, *Computer Vision–ACCV 2020*, volume 12539, pages 429–448, Cham, 2020. Springer International Publishing. doi: 10.1007/978-3-030-68238-5_32.

[17] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. https://github.com/bethgelab/foolbox, 2017.

[18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[19] Jiawei Su, Danilo Vasconcellos Vargas, and Sakurai Kouichi. One pixel attack for fooling deep neural networks. 2017. URL https://arxiv.org/abs/1710.08864.

[20] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[21] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[22] Chaowei Xiang and Prateek Mittal. Detectorguard: Provably securing object detectors against localized patch hiding attacks. *arXiv preprint arXiv:2102.02956*, 2021. doi: 10.48550/arXiv.2102.02956.

[23] Yingyu Zhang, Yang Song, Hairong Qi, Yingjie Xia, Cho-Jui Hsieh, and Le Song. Detecting adversarial attacks via data distribution discrepancy. In *International Conference on Learning Representations*, 2019.